Administración de Base de Datos

TESH
HUIXQUILUCAN

Creación de una base de datos

```
1  CREATE DATABASE universidad;
```

Selecciona base de datos

```
1  USE universidad;
```

Crea tablas(Departamentos, profesores, Cursos, Estudiantes, RegistroCursos)

```
1  CREATE TABLE Departamentos (
2      DepartamentoID INT PRIMARY KEY,
3      Nombre VARCHAR(100),
4      Descripcion TEXT
5  );
```

```
1  CREATE TABLE Profesores (
2      ProfesorID INT PRIMARY KEY,
3      Nombre VARCHAR(100),
4      Apellido VARCHAR(100),
5      Titulo VARCHAR(50),
6      DepartamentoID INT,
7      FOREIGN KEY (DepartamentoID) REFERENCES Departamentos(DepartamentoID)
8  );
```

```
1  CREATE TABLE Cursos (
2      CursoID INT PRIMARY KEY,
3      Nombre VARCHAR(100),
4      Creditos INT,
5      ProfesorID INT,
6      DepartamentoID INT,
7      FOREIGN KEY (ProfesorID) REFERENCES Profesores(ProfesorID),
8      FOREIGN KEY (DepartamentoID) REFERENCES Departamentos(DepartamentoID)
9  );
```

```
1  CREATE TABLE Estudiantes (
2      EstudianteID INT PRIMARY KEY,
3      Nombre VARCHAR(100),
4      Apellido VARCHAR(100),
5      DepartamentoID INT,
6      FOREIGN KEY (DepartamentoID) REFERENCES Departamentos(DepartamentoID)
7  );
```

```
CREATE TABLE RegistroCursos (
    ID_RegistroCursos INT AUTO_INCREMENT PRIMARY KEY,
    EstudianteID INT,
    CursoID INT,
    Semestre VARCHAR(20),
    Calificacion CHAR(2),
    FOREIGN KEY (EstudianteID) REFERENCES Estudiantes(EstudianteID),
    FOREIGN KEY (CursoID) REFERENCES Cursos(CursoID)
);
```

Inserción de datos para las tablas

```
INSERT INTO Departamentos (DepartamentoID, Nombre, Descripcion)
VALUES
(1, 'Ciencias de la Computación', 'Departamento dedicado a la enseñanza e
investigación en el campo de la informática'),
 (2, 'Matemáticas', 'Departamento enfocado en matemáticas aplicadas y puras'),
(3, 'Literatura', 'Departamento que estudia las diversas formas literarias y su
evolución a lo largo del tiempo');
```

```
INSERT INTO Profesores (ProfesorID, Nombre, Apellido, Titulo, DepartamentoID)
VALUES (1, 'Ana', 'García', 'Dr.', 1),
(2, 'Luis', 'Molina', 'Dr.', 2),
(3, 'Carlos', 'Pérez', 'Mg.', 1),
 (4, 'María', 'Fernández', 'Dr.', 3);
```

```
INSERT INTO Cursos (CursoID, Nombre, Creditos, ProfesorID, DepartamentoID)
VALUES
 (1, 'Introducción a la programación', 4, 1, 1),
(2, 'Estructuras de datos', 3, 3, 1),
(3, 'Álgebra Lineal', 3, 2, 2),
 (4, 'Literatura Española', 2, 4, 3);
```

```
INSERT INTO Estudiantes (EstudianteID, Nombre, Apellido, DepartamentoID)
VALUES
(1, 'Jorge', 'López', 1),
 (2, 'Isabel', 'Martinez', 2),
(3, 'Sofía', 'Rojas', 3),
(4, 'Pedro', 'Gómez', 1);
```

```
INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion)
VALUES
 (1, 1, '2024A', 'A'),
 (1, 2, '2024A', 'B'),
 (2, 3, '2024A', 'A'),
(3, 4, '2024A', 'A'),
(4, 1, '2024A', 'C');
```

Transacción para inscribir a un estudiante en varios cursos
Supongamos que queremos registrar a un estudiante en varios cursos al mismo tiempo, asegurándonos de que todas las inscripciones se realicen con éxito o que ninguna se realice (todo o nada).

```
START TRANSACTION;

INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion)
VALUES (4, 2, '2024A', 'A');
INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion)
 VALUES (4, 3, '2024A', 'B');

-- Suponiendo que todo está correcto, confirmamos los cambios
COMMIT;
```

Si hubiera algún problema durante la inserción, por ejemplo, un conflicto de clave primaria o una violación de clave foránea, podrías usar un ROLLBACK para deshacer todas las operaciones desde el inicio de la transacción:

```
START TRANSACTION;

INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion) VALUES (4,
2, '2024A', 'A');
INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion) VALUES (4,
3, '2024A', 'B');

-- Si algo falla, deshacemos todo
ROLLBACK;
```

Transacción para actualizar datos de un curso y registrar nuevos estudiantes
Supongamos que quieres actualizar el número de créditos de un curso y al mismo tiempo registrar nuevos estudiantes en ese curso. Quieres asegurarte de que ambas operaciones se ejecuten juntas:

```
START TRANSACTION;

```

```
UPDATE Cursos SET Creditos = 5 WHERE CursoID = 2;
INSERT INTO Estudiantes (EstudianteID, Nombre, Apellido, DepartamentoID)
VALUES (5, 'Laura', 'Sanchez', 1);
INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion)
VALUES (5, 2, '2024A', 'A');

COMMIT;
```

En este caso, si el UPDATE o cualquiera de los INSERT falla por alguna razón, puedes decidir usar ROLLBACK para revertir todas las modificaciones:

```
START TRANSACTION;

UPDATE Cursos SET Creditos = 5 WHERE CursoID = 2;
INSERT INTO Estudiantes (EstudianteID, Nombre, Apellido, DepartamentoID)
VALUES (5, 'Laura', 'Sanchez', 1);
INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion)
VALUES (5, 2, '2024A', 'A');

-- En caso de error
ROLLBACK;
```

Uso de TRY CATCH (para manejar errores en algunos RDBMS)
Algunos sistemas de gestión de bases de datos soportan estructuras de control de errores como TRY CATCH. Aquí te muestro cómo se vería en esos sistemas:

```
BEGIN TRY
    START TRANSACTION;

    UPDATE Cursos SET Creditos = 5 WHERE CursoID = 2;
    INSERT INTO Estudiantes (EstudianteID, Nombre, Apellido, DepartamentoID) VALUES
(5, 'Laura', 'Sanchez', 1);
    INSERT INTO RegistroCursos (EstudianteID, CursoID, Semestre, Calificacion) VALUES
(5, 2, '2024A', 'A');

    COMMIT;
END TRY
BEGIN CATCH
    ROLLBACK;
    -- Aquí puedes manejar el error, como loguearlo o informar al usuario
END CATCH;
```

Supongamos que:

El estudiante con ID 1 pasa del CursoID 1 al CursoID 3.
El estudiante con ID 2 pasa del CursoID 3 al CursoID 1.
El curso con ID 2 cambia su profesor al ProfesorID 2.

```
DELIMITER //

CREATE PROCEDURE CambiarEstudiantesYCursos()
BEGIN
    DECLARE exit handler for sqlexception
    BEGIN
        -- En caso de error, se revertirán los cambios
        ROLLBACK;
        RESIGNAL;
    END;

    START TRANSACTION;

    -- Cambiar a los estudiantes de curso
    UPDATE RegistroCursos SET CursoID = 3 WHERE EstudianteID = 1 AND CursoID = 1;
    UPDATE RegistroCursos SET CursoID = 1 WHERE EstudianteID = 2 AND CursoID = 3;

    -- Cambiar el profesor de un curso
    UPDATE Cursos SET ProfesorID = 2 WHERE CursoID = 2;

    -- Si se ejecuta sin errores, hacer commit de la transacción
    COMMIT;
END//

DELIMITER ;

-- Ejecutamos el procedimiento
CALL CambiarEstudiantesYCursos();
```